1.) A child is running up a staircase with n steps and can hop either 1 step, 2 steps, or 3 steps at a time. Implement a method (that uses recursion) to count how many possible ways the child can run up the stairs.

Are there potential errors to check for? If so, what?
If n < 0 return 0 OR
If n <= 0, return 0 depending on base case.

What is the base case?
If n == 0, return 1 OR
If n == 1, return 1 elif n == 2, return 2 elif n == 3, return 4

```
public int countWays(int n) {

       if(n < 0){
              return 0;
       }else if (n == 0){
              return 1;
       }else {
              return countWays(n-1) + countWays(n-2) +
              countWays(n-3)
       }
}
```

2.) Given a 2d grid map of 1's (land) and 0's (water), count the number of islands. An island is surrounded by water and is formed by connecting adjacent lands horizontally or vertically. You may assume all four edges of the grid are all surrounded by water. You may alter the board in order to reach your solution.

The "starter" method is given for you – please implement the recursive "helper" method.

Ex:

```
1 1 0 0 0
1 1 0 0 0     →    Returns 3, as there are 3 distinct islands.
0 0 1 0 0
0 0 0 1 1
```

What are potential errors to check for?

While checking for islands in the recursive method, ensure your recursive method does not go out of bounds.

What is the base case?

If n == '0', you have not found an island. Do not recurse.

What is the "numIslands" method implemented on the next page doing (how does it relate to the recursive method you need to implement)?

It looks through every char in the grid and puts that character into the recursive method. This means that the recursive method must determine if the character is part of an island or not, and return only 1 for each island that is found.

```
//return number of islands in the grid
public int numIslands(char[][] grid) {
     if(grid == null || grid.length == 0 || grid[0].length == 0){
          return 0;
     }
     int result = 0;
     for(int i = 0; i < grid.length; i++){
          for(int j = 0; j < grid[i].length; j++){
               result += helper(grid, i, j);
          }
     }
     return result;
}


// implement this recursive method
private int helper(char[][] grid, int row, int col){
     if(r < 0 || r >= grid.length || c < 0 || c >=
          grid[0].length || grid[r][c] == '0'){
          return 0;
     }
     grid[r][c] = '0';//set current to 0 – don’t want to recount
     helper(grid, r+1, c);
     helper(grid, r-1, c);
     helper(grid, r, c+1);
     helper(grid, r, c-1);
     return 1; //return only 1 for the entire island
}
```